Remarks

This Amendment responds to the final Office Action ("the Action") mailed March 13, 2007. Reconsideration of the application is respectfully requested in view of the foregoing amendments and following remarks. Claims 1-36 are pending in the application. No claims have been canceled. No claims have been allowed. Claims 1, 21, 25, 27, 29, and 32 are independent.

Cited Art

U.S. Patent Application No. 11/330,053 (Pub. No. 2006/0129994) to Srivastava et al. ("the '053 application") is entitled "Method and Apparatus for Prioritizing Software Tests."

The February 2002 publication by Srivastava et al. is entitled "Effectively Prioritizing Tests in Development Environment" ("Srivastava").

Amendments

Editorial amendments have been made to claims 1, 21, 27, 29, and 32. No new matter has been added.

Previously Addressed Objections and Rejections

In the previous Amendment, filed December 18, 2006, Applicants responded to an objection to claim 29 and to rejections under 35 U.S.C. §§ 101 and 112. As the current Action does not mention these particular objection and rejections, Applicants assume that the Examiner found the arguments persuasive and has withdrawn the objection and the rejections. If this is not the case, and if any of these rejections and objection remain, Applicants request that the Examiner inform Applicants as soon as possible.

Claim Objections

The Action objects to informalities in claims 1 and 27. In particular, the Examiner reads the language of claims 1 and 27, specifically the language "requesting . . . a . . . request" as "awkward." [Action, at §2, page 2.] While the Applicants do not necessarily agree that the language as presented was awkward, in the interest of expediting prosecution, claims 1 and 27 have been amended. As such,

the claims now read, "*requesting*, via an application programming interface . . . , *a dependency collection, by sending to the application programming interface a system dependency creation request* . . . ."

Applicants believe that, with these amendments, the language of claims 1 and 27 should no longer appear awkward. Applicants thus request that the objections to claims 1 and 27 be withdrawn.

Rejections Under 35 U.S.C. § 102 over Srivastava

The Action rejects claims 1-36 under 35 U.S.C. § 102(b) as being anticipated by Srivastava. Applicants respectfully submit the claims in their present form are allowable over the cited art. For a 102(b) rejection to be proper, the cited art must show each and every element as set forth in a claim. (*See* MPEP § 2131.01.) However, the cited art does not describe each and every element. Accordingly, applicants request that all rejections be withdrawn. Claims 1, 21, 25, 27, 29, and 32 are independent.

Claim 1

Claim 1, as amended, recites, in part:

> *receiving a system definition . . . the system definition comprising metadata describing a system at a level of abstraction*;
> requesting . . . a dependency collection, by sending . . . *a system dependency creation request comprising the received system definition and a dependency request comprising a target logical abstraction*; . . . .

[Emphasis added.] An example system definition is illustrated in Figure 6. The Application describes the system definition of Figure 6 as comprising metadata describing a system at a level of abstraction:

> Figure 6 shows an example system definition file. *In this case, the system definition file is represented as an XML file 600. The abstraction levels in this example are defined as system 602, subsystem 606, and binary (file) 608.* In this example, the system definition file identifies the universe of desired dependencies by indicating the names 608 of the input binary files, and the name 608 of the XML file where the binary file dependency relationships are stored. Also, the example shows a subsystem name 606, and the name 610 of the XML file where the subsystem dependency relationships are stored. The names and arrangement of the mark-up tags in the XML files may be changed and arranged to indicate desired levels of granularity and abstractions. The dependency information is stored in XML files (e.g., 610, 614) according to the levels of abstraction of an example system.

[Application, at page 13, line 25 to page 14, line 7; emphasis added.]

The "receiving" and "requesting" language of claim 1 recite distinct actions, and should not be interpreted as the same action. Applicants note that in Section 5(A), pages 12-15, the Action alleges, inter alia, that the "requesting" and "receiving" language of claim 1 are part of the same action. For example, the Action interprets the "receiving a system definition" language as "being the same as (or subsumed in) the step of requesting which also comprises reception of a system definition." [Action, at § 5(A), page 12.]

Applicants respectfully submit that this is an incorrect interpretation of the claim. In particular, the "requsting" language of the claim in fact *utilizes* a request comprising a system definition, it *does not* request such a system definition, since it has already received the system definition. In the interest of expediting allowance, however, Applicants have amended the claim to more clearly show the relationship between the "receiving" and "requesting" language.

In particular, the claim now recites "requesting . . . *a dependency collection*." The claim also clearly recites that the request is done "*by sending . . . a system dependency creation request comprising the received system definition* and a dependency request comprising a target logical abstraction . . . ." Thus, by the language of claim 1, the "requesting" action utilizes a "received system definition." This is the "system definition" which is received in the previously-recited "receiving a system definition" language. The end result of this requesting is then "a dependency collection" rather than the "request" itself, which appears to be the way the language was previously interpreted in the Action. By these amendments, the claim language should be sufficient to demonstrate that the two actions are distinct and should not be interpreted as the same action.

Srivastava performs test prioritization through direct analysis of binary files, and thus does not describe a "system definition comprising metadata describing a system at a level of abstraction" as recited in claim 1. In its rejection of claim 1, the Action alleges that the "system definition" of claim 1 is described at, among other locations, Figure 1, and pages 2 and 3 of Srivastava. However, as the cited passages make clear, Srivastava's "Echelon" system uses executable binary files, rather than system definition files, to perform its analysis. For example, Figure 1 illustrates "Old Binary" and "New Binary" as input into the "Binary Change Analysis" module. Srivastava makes this clear in its description accompanying Figure 1:

> Echelon takes as input two versions of a program in binary form along with the test coverage information of the old binary and produces a prioritized list of the tests, as well as a list of the modified and new blocks (or source files) that may not be executed by any existing test. Echelon accomplishes this task in three steps illustrated in Figure 1.
>
> In the first step, Echelon uses BMAT to find a matching block the old binary for each block in the new binary.

[Srivastava, at page 3, left col., para. 5 to page 3, right col., top para.]

Thus, the system of Srivastava is directed toward direct analysis of binary executable files rather than usage of a "system definition" as is recited in claim 1. Furthermore, by operating directly on binary files, Srivastava demonstrates that it does not utilize a system definition which "compris[es] metadata describing a system at a level of abstraction," as recited in claim 1, since it avoids abstraction by going straight to the binary files.

For at least these reasons, Srivastava does not teach or suggest the above-recited language of claim 1 and thus Srivastava does not describe each and every element of claim 1. Claim 1, as well as claims 2-20, which depend from claim 1, are thus allowable and applicants request their allowance.

Claim 21

Claim 21, as amended, recites, in part:

> determining dependency information about binary files;
>
> propagating dependency information about binary files to determine subsystem dependency information for a subsystem of a system;
>
> propagating the subsystem dependency information to determine system dependency information for the system; . . . .

For example, the Application describes, in the course of describing an exemplary process for exposing binary dependencies illustrated at Figure 5, an exemplary process for propagating dependencies:

> *At 506, relationships between binary dependency files are propagated to reflect dependencies between binary files. Dependency relationships are built by connecting all the exit points of a binary dependency file to the corresponding entry points of the binary dependency file where control is transferred.* For example, as shown in Figure 9, the method 500 creates information 902 comprising binary dependencies. In this example, the information indicates a dependency between exit points and entry points. At this level of abstraction, an exit point is a binary file name 908 and an exit location 914 (e.g., BDF A, OUT1). An entry point is a binary file name 910 and an entry point 916 (e.g., BDF C, C1). *At this level of abstraction, a binary dependency 902 is an exit point, entry point pair. The method examines each binary dependency file 908, and creates the exit-entry pairs 902-906 for the binary dependency file 908.*

[Application, at page 12, lines 10-20; emphasis added.]

Srivastava, which only propagates changes found in binary files, does not "propagate[e] dependency information about binary files to determine subsystem dependency information" or "propagating the subsystem dependency information to determine system dependency information" as recited in claim 21. In its rejection of claim 21, the Action cites to the 5th paragraph of the right column of page 2. The particular cited sentence states:

> Moreover, by the time the program is available in binary form, all the changes in header files to constants, macro definitions, etc. have already been propagated to the affected procedures in the program, thus simplifying the process for determining program changes.

Applicants note first that the passage cited in the rejection does not discuss propagation performed by the system of Srivastava. Instead, the passage notes that, by focusing on the use of binaries, the system of Srivistava is able *to assume* that the effects of any changes made in particular files *have already been propagated* throughout the binary file Srivistava is analyzing, most likely during compilation. Thus, Srivistava does not describe itself using propagation along with any of its other actions, and thus should not be understood to anticipate an action of propagation, at least based on this passage.

Secondly, Applicants note that that which is being propagated in the cited passage are *changes*, not dependencies. Thus, notwithstanding the facts above, even if the passage were to be interpreted to include propagation, there is no indication given in Srivastava that this propagation is *of dependencies*.

Applicants also note that, because no dependency information is shown to be propagated in Srivastava, Srivastava does not, and cannot, describe "determine[ing] subsystem dependency information" or "determine[ing] system dependency information" from propagation lower-level dependency information, as there is no dependency information described in the first place.

Finally, Applicants respectfully disagree with the contention, in the Action, that the data shown in Tables 1 and 2 of Srivastava, along with the algorithm of Fig. 2, could show that Srivastava describes, or even teaches or suggests, the recited propagation. The two Tables describe statistics of changes made between two versions of a large program. Figure 2 shows an algorithm which is used *to prioritize test coverage* after the Echelon system of Srivastava has determined changes between two versions of a program. This is made clear from the inputs to the algorithm, which utilize a set of changed blocks which have already been determined, as well as a set of tests and the blocks affected by the tests. This is also made clear in the passage following Figure 2 which states "[a]s shown in Figure 2, Echelon uses the impacted block set for each test to prioritize the tests.] [Srivastava, at page 4, Left column, paragraph 3.] Thus, this additional information cited by the Action does not describe the propagation of information. At best, even if it were interpreted to describing the *results* of propagation, which is not the same as describing propagation itself, the cited passages would then only describe propagation of *change* information, not dependency information.

For at least these reasons, Srivastava does not teach or suggest at least the above-recited language of claim 21 and thus Srivastava does not describe each and every element of claim 21. Claim 21, as well as claims 22-24, which depend from claim 21, are thus allowable and applicants request their allowance.

Claim 25

Claim 25 recites, in part:

> means for determining binary dependencies for a defined system;
> means for propagating binary dependencies to identify binaries dependent on binaries in other subsystems; . . . .

In its rejection of claim 25, the Action, in reference to the above-quoted language, cites to a passage of Srivastava which it cited in its rejection of claim 21. Thus, for at least the reasons discussed above with respect to claim 21, Srivastava does not teach or suggest at least the above-recited language of

claim 25 and thus Srivastava does not describe each and every element of claim 25. Claim 25, as well as claim 26, which depends from claim 25, are thus allowable and applicants request their allowance.

Claim 27

Claim 27, as amended, recites, in part:

> *creating a system definition comprising metadata describing a system at a level of abstraction in response to receiving graphical user interface input*;
> *receiving a dependency information request* via graphical user interface input;
> requesting . . . a dependency collection, by sending . . . *a system dependency creation request comprising the system definition, and a target logical abstraction* . . . .

In its rejection of claim 27, the Action, in reference to the above-quoted language, cites to passages of Srivastava which it cited in its rejection of claim 1. Thus, for at least the reasons discussed above with respect to claim 1, Srivastava does not teach or suggest at least the above-recited language of claim 27 and thus Srivastava does not describe each and every element of claim 27. Claim 27, as well as claim 28, which depends from claim 27, are thus allowable and applicants request their allowance.

Claim 29

Claim 29, as amended, recites, in part:

> propagating dependency information about binary files to determine subsystem dependency information for a subsystem of a system; and
> propagating subsystem dependency information to determine system dependency information for the system; . . . .

In its rejection of claim 27, the Action, in reference to the above-quoted language, cites to its rejection of claim 21. Thus, for at least the reasons discussed above with respect to claim 21, Srivastava does not teach or suggest at least the above-recited language of claim 29 and thus Srivastava does not describe each and every element of claim 29. Claim 29, as well as claims 30 and 31, which depends from claim 29, are thus allowable and applicants request their allowance.

Claim 32

Claim 32, as amended, recites, in part:

> means for accepting a system definition comprising metadata describing a system at a level of abstraction and indicating binary files in plural subsystems . . . .

In its rejection of claim 32, the Action, in reference to the above-quoted language, cites to passages of Srivastava which it cited in its rejection of claim 1. Thus, for at least the reasons discussed above with respect to claim 1, Srivastava does not teach or suggest at least the above-recited language of claim 32 and thus Srivastava does not describe each and every element of claim 32. Claim 32, as well as claims 33-36, which depend from claim 32, are thus allowable and applicants request their allowance.

Double Patenting Rejection

The Action provisionally rejects claims 21, 25, and 29 under the judicially created doctrine of obviousness-type double patenting as being unpatentable over the '053 application in view of Srivastava. Applicants respectfully submit the claims in their present form are patently distinct over the cited art. To establish a *prima facie* case of obviousness, three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations. (MPEP § 2142.)

In the course of the provisional double patenting rejection, the Action admits that '053 application claims do not recite a number of features, including:

> determining dependency information about binary files, propagating such information to determine subsystem and system dependency, marking changed and unchanged logical abstraction to prioritize tests.

[Action, at § 7, page 18.] Instead, the Action finds these features in Srivastava, relying on the same and similar passages as used in the rejections of claims 21, 25, and 29 under 35 U.S.C. §102, which were discussed above.

Applicants respectfully note that, for at least the reasons given above with respect to claims 21, 25, and 29, Srivastava does not teach or suggest at least one of the features of each claim. For example, Srivastava does not teach or suggest "propagating dependency information about binary files to determine subsystem dependency information for a subsystem of a system" and "propagating the subsystem dependency information to determine system dependency information for the system" as recited in claim 21. Because language from each of claims 21, 25, and 29 is found neither in the '053

application claims nor in Srivastava, such language is not taught or suggested by the combination of the two.

For at least this reason, the Action fails to make a establish a *prima facie* case of obviousness over the '053 Application in view of Srivastava. Accordingly, applicants request that the provisional double patenting rejection be withdrawn.

Request for Interview

If any issues remain, the Examiner is formally requested to contact the undersigned attorney prior to issuance of the next Office Action in order to arrange a telephonic interview. It is believed that a brief discussion of the merits of the present application may expedite prosecution. Applicants submit the foregoing formal Amendment so that the Examiner may fully evaluate Applicants' position, thereby enabling the interview to be more focused.

This request is being submitted under MPEP § 713.01, which indicates that an interview may be arranged in advance by a written request.

Conclusion

The claims in their present form should now be allowable. Such action is respectfully requested.

Respectfully submitted,

KLARQUIST SPARKMAN, LLP

One World Trade Center, Suite 1600
121 S.W. Salmon Street
Portland, Oregon 97204
Telephone: (503) 595-5300
Facsimile: (503) 595-5301

By _____
Stephen A. Wight
Registration No. 37,759